# Event Registration Desktop Application

## 1 Required Software Engineering Tools

- Eclipse Neon Modeling Tools (http://www.eclipse.org/downloads/packages/eclipse-modeling-tools/neonr)
- Umple Eclipse plugin 1.22.0.5146 (or later) – to be installed via running Eclipse (http://cruise.eecs.uottawa.ca/org.cruise.umple.eclipse.plugin.update.site)
- Umple Online (http://cruise.eecs.uottawa.ca/umpleonline/)
- JUnit 4 (included with Eclipse)
- XStream libraries (provided in myCourses)
- JDatePicker library (provided in myCourses)
- *xmlpull-1.1.3.1.jar*, *xpp3_min-1.1.4c.jar* (provided in myCourses)

Go to the above link and download the Eclipse Modeling Tools for your operating system, and install Eclipse.

To install Umple, Go to Help → Update New Software, then click on the "Add" button in the top right corner and give http://cruise.eecs.uottawa.ca/org.cruise.umple.eclipse.plugin.update.site as URL for the update site (and e.g. "Umple" as a name). Then select all Umple plugins from the list in the middle and click on Finish. You may need to accept some licenses and restart your Eclipse workbench after installation is completed.

## 2 Description of Event Registration Desktop Application

Typically, this description is elicited from stakeholders (e.g., potential customers). However, for our purposes, we will assume that it is provided as follows:

- The Event Registration application shall provide the ability to add a participant by specifying the participant's name.
- The Event Registration application shall provide the ability to add an event by specifying the event's name, date, start time, and end time.
- The Event Registration application shall provide the ability to register a participant to an event.

## 3 Tasks of the Assignment

To complete this assignment, you need to follow the tutorial at
https://ecse321-winter2017-mcgill.github.io/EventRegistration-Tutorials/#_java_swing

In addition, short demonstration videos are also provided in MyCourses. Note that the source code listed in the textual version tutorial is your primary reference (i.e. it is maintained continuously).

The main steps you need to complete are the following (section numbers refer to the online document at the above link while zip files are provided in myCourses):

- **Sec. 1.2**: Define the domain model with Umple Online and generate Java code with the Umple Eclipse plugin. See also *EventRegistrationDesktop-DomainModel.html* in the *D03 Domain Model.zip* file to find a short video demonstration.
- **Sec 1.3:** test the persistence functionality of XStream with JUnit 4. See *EventRegistrationDesktop-TestPersistence.html* in the *D04 Test Persistence.zip* file
- **Sec 1.4**: Implement one test for one method of the controller of the Event Registration Desktop Application and then implement this method of the controller following the Test-Driven Development paradigm. See *EventRegistrationDesktop-Controller.html* in *D05 Controller.zip*.
- **Sec 1.5**: Implement further tests for the same method following the Test-Driven Development paradigm to take invalid inputs into account and then add input validation to this method of the controller. See also *EventRegistrationDesktop-ControllerInputValidation.html* in the *D06 Controller Input Validation.zip* file.
- **Sec 1.6**: Implement an initial User Interface of the Event Registration Desktop Application with the help of the Java Swing Framework. See *EventRegistrationDesktop-View.html* in *D07 View.zip.*
- **Sec 1.7**: Use the Test-Driven Development paradigm to implement the "everything is ok" scenarios for the remaining methods of the controller of the Event Registration Desktop Application. See *EventRegistrationDesktop-ControllerPart2.html* in *D08 Controller Part 2.zip.*
- **Sec 1.8**: Use the Test-Driven Development paradigm to implement the scenarios with invalid input for the remaining methods of the controller of the Event Registration Desktop Application. See *EventRegistrationDesktop-ControllerInputValidationPart2.html* in the *D09 Controller Input Validation Part 2.zip.*
- **Sec 1.9**: Implement the User Interface of the remaining functionality of the Event Registration Desktop Application with the help of the Java Swing Framework. See *EventRegistrationDesktop-ViewPart2.html* in the *D10 View Part 2.zip.*

## Submission

This assignment is to be done in teams of **TWO** students. You are required to **sign up to one of the assignment groups** in myCourses as soon as possible but definitely before **Wednesday, January 18th, 2017**.

Your team is required to hand in a ***single zip file*** of the Eclipse project with your implementation by ***Wednesday, February 1, 2017 at 23:30***. To create the zip file, use the *Export* feature of Eclipse to create an *Archive File* (Export – General – Archive File) of the project. If you realize that you need to make changes to your submission, do not resubmit only the file(s) that have changed, but rather resubmit another complete zip file.

**Note that this is only part 1 of 3 parts.** The second and third part of assignment #1 will be posted on January 23rd. Each part is about the same amount of work and all parts are due on **Wednesday, February 1st, 2017 at 23:30**. Do not wait until the very end to complete this assignment. Start early.

Each team member must make contributions to the assignment. A team member who does not contribute to the assignment receives a mark of 0 for the assignment. A team member may optionally email a confidential statement of work to the instructor before the due date of the assignment. A statement of work first lists in point form the parts of the assignment to which the team member contributed. In addition, the statement of work also describes whether the work load was distributed fairly evenly among the team members. A statement of work may be used to adjust the mark of a team member who is not contributing sufficiently to the assignment. It is not necessary to send a statement of work, if a team distributed the work for the assignment fairly evenly and each team member contributed sufficiently.

## Marking Scheme

| ***Part of Assignment*** | ***Marks*** |
|---|---|
| Domain Model in Umple | 15 |
| Use of Code Generated from Domain Model | 5 |
| Adherence to Model-View-Controller Pattern | 20 |
| JUnit Tests | 10 |
| Implementation of Functionality: | 40 |
| a) Add Participant | 10/40 |
| b) Add Event | 10/40 |
| c) Register | 10/40 |
| d) Persistence | 10/40 |
| Implementation of Validation Checks for: | 10 |
| a) Add Participant | 2/10 |
| b) Add Event | 4/10 |
| c) Register | 4/10 |
| Total Marks: | 100 |
| The total mark may be adjusted based on the actual contributions of a team member to the assignment. | |